纵横互联网系列丛书

百变金刚

# JSP网络编程实例


雷富强
王志勇 编著
吴斌 监制
程峰 策划

中国电力出版社

纵横互联网系列丛书

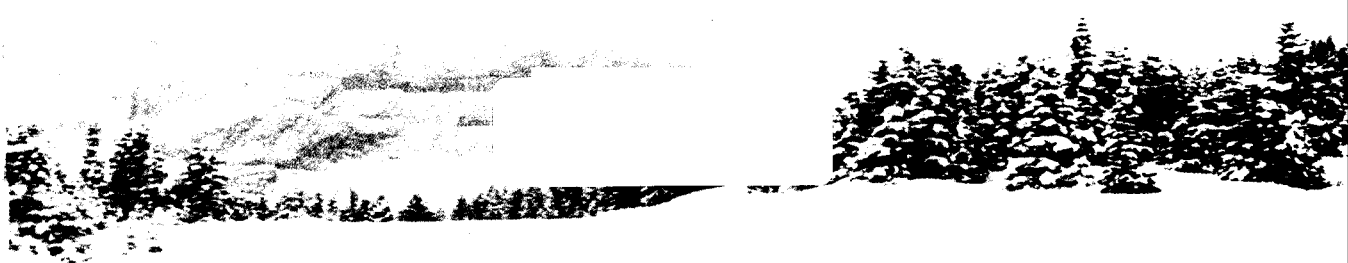


# 百变金刚

# JSP网络编程实例

雷富强
王志勇
编著
吴斌
监制
程峰
策划





中国电力出版社

## 内 容 提 要

本书以实例的方式，由浅入深地带您进入 JSP 的精彩世界。JSP 是由 SUN 公司于 1999 年推出的一种与平台无关的新计算机技术，非常适合 Web 应用程序的开发。JSP 集极高的运行效率、较短的开发周期、超强的扩展能力、完全开放的技术标准等众多完美特性于一身，迅速成为光彩夺目的技术新星。本书通过 22 个 JSP 网络编程实例,详细讲解了这一项技术的技巧以及如何利用这项技术建立先进、安全、跨平台的商业网站。

本书所附光盘中包含书中提及的实例的源代码，这些程序均来自网站建设实战，经过专门的检测调试，可以即学即用。本书适合 JSP 的初学者，同时也可作为中高级读者、高等院校师生和网络工程师的参考书。

中国电力出版社出版、发行

（北京三里河路 6 号 100044 http://www.infopower.com.cn）

汇鑫印务有限公司印刷

各地新华书店经售

*

ISBN 7-900038-97-3/TP · 84

2002 年 7 月第一版 2002 年 7 月北京第一次印刷

787 毫米×1092 毫米 16 开本 20 印张 446 千字

定价 32.00 元

# 丛书前言

网络的诞生，改变了人们使用计算机的方式；而 Internet 的出现，又改变了人们使用网络的方式。Internet 的快速发展，代表着人类历史上另一场革命的开始。曾有人断言：假如有一天，21 世纪的人类文明就像玛雅文化一样神奇地消失的话，那么，当后世的考古学家在发掘这段历史的时候，一定会把 Internet 当做是一场瘟疫。因为，它在极短的时间内就充斥了各种各样的媒体。虽然，这只是笑谈，但从中不难看出 Internet 对人类的生活所造成的影响是何等深远而广泛！

网络时代已经来临！缤纷多彩的网络世界改变着人们的生活，特别是当 WWW 日益普及之际。人们对信息资源的渴求是无止境的，开发者们则在不断地扩充超媒体语言的能力。从静态文本到静态图像，从静态图像再到动态图像，随后又加入了声音、影像、三维动画等，网络世界从此变得生机盎然。

目前，图书市场上出现了很多介绍 Internet 编程方面的书籍。但是，这些书籍或者过于简单，让读者觉得收获甚微；或者因过于专业化而未能兼顾初级水平的用户；或者着重于理论，而缺乏具体的实例剖析；或者理论与实例相背离，没有起到应有的效果。因考虑到很多用户并没有太多的实际编程经验，故本丛书力图通过实例，系统全面地介绍网络编程各个方面的基本原理和开发技巧，以尽可能地满足不同层次读者的要求，使读者能通过本丛书切实掌握一门网络编程语言的技巧和方法，可以自己开发出功能强大的网站。

本丛书以讲解和分析实例为主。丛书中的每本书籍，从最简单的例子到高级的编程技巧，都有所涉猎。希望通过学习本丛书，无论是网络开发的新手，还是经验丰富的“老鸟”，都可以从中获益。

本丛书的讲解全面细致，举例典型实用，使读者可以迅速走入网络编程的一片新天地，继而可以熟练地使用一门网络编程语言进行开发。同时，本丛书为专业的开发人员提供了详尽的参考，有利于进一步提高编程的水平，掌握更科学的编程技巧。本丛书的实例都是按照众多高手们从入门到精通的学习过程来编排的，以方便读者循序渐进地学习。总之，编写本丛书的目的就是为广大读者提供一套系统全面的网络编程的实用教程，希望本丛书的出版可以对大家的实际工作有所帮助。

编写本丛书可不是一件轻松的事情，其中凝聚了太多人的努力和无私的奉献。首先，要感谢每一本书的作者，本丛书是它们渊博知识的凝聚，也是它们心血的结晶。还要感谢编校人员，它们认真细致的工作作风使得本丛书尽量少出错。最后还要感谢在我身后一直给予支持的家人和朋友们，有了你们的理解和支持，这栋大厦才得以构建。

对于本丛书中的每一本书，由于每位作者的学识、能力有限，书中疏漏或错误之处在所难免，敬请广大读者批评指正。

编　者

# 前　言

随着技术的发展，大量的软件开发工作已经从客户端迁移到了服务器端。由客户端处理复杂的数据已不适应主要的企业应用程序。而是应用程序本身驻留在服务器上，将数据格式化为适合用户浏览的 Web 页面，并且处理用户填写的 Web 表单的响应。如果要开发这样的 Web 应用程序，需要能够方便地产生大量动态 HTML 页面、访问数据库资源、具有优越的性能和稳定性。

由 Sun 微电子系统公司在 1999 年 6 月推出的 JSP（Java Server Pages）是一种完全与平台无关的开发新技术，非常适合于 Web 应用程序的开发。JSP 集成了极高的运行效率、较短的开发周期、超强的扩展能力、完全开放的技术标准、自由的开发方式等众多完美特性于一身，迅速成为光彩夺目的技术明星。经过几年的不断发展和完善，JSP 现在已经十分成熟，它基于 Java Servlet 以及整个 Java 体系的 Web 开发技术，是 Servlet2.1 API 的扩展。利用这一技术可以建立先进、安全和跨平台的动态网站。

JSP 的结构与 ASP 非常相似，实际上 JSP 也参考借鉴了 ASP 的许多重要思想和有价值的东西。所不同的是 ASP 一般只应用于 Windows 平台，而 JSP 得利于 Java 的跨平台性，并受到各方大力支持，因此可以运行在大多数 Web Server 上。JSP 借用了 ASP 的许多优点，如 Session、Application 等对象。同时 JSP 使用的是灵活而强大的 Java 语言，而不是低效的脚本语言。

J2EE（Java2 platform Enterprise Edition）则是 Sun 为企业级应用建立的高水准的框架结构，获得了各大开发平台提供商的积极支持，并已成为企业应用开发的主流产品。JSP 技术则是 J2EE 平台上是一个关键性的组件。

我们选择 JSP 的原因有下面几个理由：

（1）一处编写随处运行。这是一个程序员多年的梦想，为了在不同的平台间运行，许多程序员一行行的重写代码，在枯燥繁琐中奋斗。在这一点上 Java 已经给了我们最完美的答案，它做的要比 PHP 更加出色，除了系统之外，你精心编写的代码不用丝毫更改。

（2）系统的多台平支持。几乎在所有平台上都实现了一致的 Java Servlet/JSP 的 Web Server/Application Server 系统，可以让你在任意环境中开发，在任意环境中进行系统部署，在任意环境中扩展。相比 ASP+PHP，JSP 在这点上的优越性是显而易见的。

（3）强大的可伸缩性。从只有一个小的 jar 文件就可以运行的 Servlet/JSP，到由多台服务器进行集群和负载均衡，以至到多台 Application 进行事务处理、消息处理，或者从一台服务器扩展到无数台服务器，都显示了 Java 强大的伸缩性和生命力。

（4）多样化和功能强大的开发工具支持。这一点与 ASP 很相似，JSP 已经有了许多非常优秀的开发工具，而且其中很多可以免费得到，并且可以顺利地运行于多种平台之下。

如何掌握 JSP 呢？当然最好的方法莫过于通过实际的例子来学习，本书将以实例方式由浅入深地带你进入 JSP 的精彩世界！

由于作者水平有限、时间仓促，书中缺点、错误在所难免，敬请读者批评指正。

作　者

2002 年 3 月

# 目　录

楔　子
丛书前言
前　言
实例 1　JSP 初体验 ····· 1
实例 2　JSP 基本语法 ABC ····· 21
实例 3　Hello World! ····· 56
实例 4　向你问好 ····· 61
实例 5　猜数字游戏 ····· 69
实例 6　获取个人信息 ····· 77
实例 7　查找你的 E-mail 地址 ····· 83
实例 8　使用 JavaMail 发送电子邮件 ····· 104
实例 9　一份耕耘，两份收获 ····· 111
实例 10　使用自定义的 Bean 组件 ····· 115
实例 11　在线投票 ····· 132
实例 12　JSP 简单的异常处理 ····· 141
实例 13　井字游戏 ····· 145
实例 14　JSP 的文件操作 ····· 158
实例 15　JSP 与 SVG 技术 ····· 170
实例 16　JSP 的数据库操作 ····· 176
实例 17　网页中注册的实现 ····· 194
实例 18　使用 Tag 库 ····· 216
实例 19　网上购物车 ····· 229
实例 20　留言簿 ····· 250
实例 21　使用 Applet 显示波动文字效果 ····· 292
实例 22　使用 Applet 显示走马灯文字 ····· 299

# 实例 1　JSP 初体验

## ☆ 实例目标

随着互联网的快速发展，附加在互联网上的功能越来越多样化。网页的设计，也已从最初的静态页面，发展到可以与使用者互动，并可以加入各种对象使之更富有变化；更进一步的是，将原本在可客户端执行的网页，提升至服务器端来执行，大大提高了执行效率，也增加了网页处理复杂资料和数据的能力。

自从 1995 年 Java 诞生以来，其应用及发展都出乎人们的预料。目前世界上已有近 200 万 Java 开发人员，约 8000 多万台机器运行着 Java 应用程序，还有众多的软件开发公司进行着 Java 应用程序的开发。在未来，将会有更多的与 Java 相关的话题产生，21 世纪将是 Java 的世纪！

JSP（Java Server Pages，Java 服务器页面）是由 Sun Microsystem 公司所开发出来的一种新规格标准 Web 服务器端的开发环境，利用它可以产生和运行动态的、交互的、高性能的 Web 服务应用程序。它是一种 HTML 内嵌式解释语言，是建立动态网站的最优秀解决方案之一，同时对其他的编程语言具有良好的兼容性，其最重要的优点是可在服务器端运行。此外，作为解释语言和流行的开发工具，JSP 保持了其传统的灵活性和简易性。

与微软的 ASP（Active Server Page，动态服务器页面）相比，SUN 公司的 JSP 具有更好的系统兼容性，它几乎可以在所有的系统平台上运行。图 1-1 显示了使用 UltraEdit 编辑的 JSP 文件。UltraEdit 具有突出显示的功能，适合一般 JSP 用户作为开发工具。

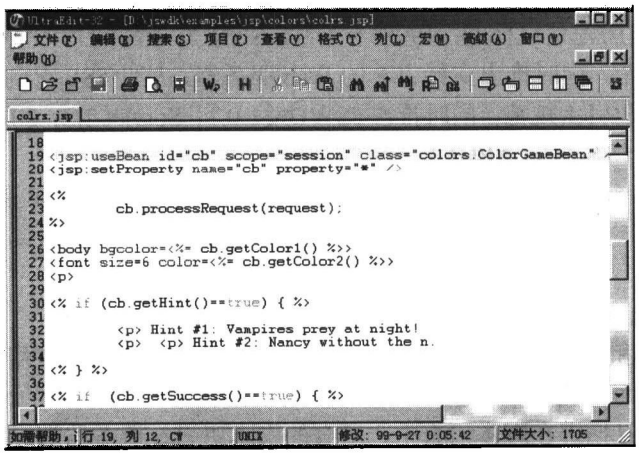
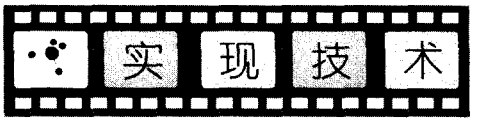



图 1-1　UltraEdit 编辑软件下的 JSP 文件

本实例主要讲述 JSP 的入门技术以及 JSP 的配置和安装，还将 JSP 与 ASP、CGI 等动态网页编程技术作了一个对比。以使读者对 JSP 有更加全面的了解。




1. JSP 概况

（1）JSP 基本概念。JSP 是用来编写动态的 Web 应用程序的开发环境，它支持多平台和多 Web 服务器，能够更有效的把服务器端的 Java 程序和静态的 HTML 页面，即“所见即所得”合并起来。向基本的 HTML 网页（即后缀名为 htm.和 html 的文件）中添加 JSP 标记就成为 JSP 程序，将其命名为后缀名为.jsp 的文件。JSP 执行.jsp 程序时，HTML 网页中的 JSP 程序部分由服务器端处理，而用户端只需用浏览器作 html 或 htm 文件处理，就得到了 HTML+JSP 的执行效果。因此，JSP 在客户端不需要做太多改动的情况下，扩展了 HTML 的功能。程序员通过使用 JSP 来突破静态网页 HTML 功能上的局限，从而给使用者带来一个功能强大的动态网页。当然，动态网页的设计语言也有许多种，而 JSP 则由于其简便、跨平台和可重用等优点得到了迅速发展，已经成为越来越多网页设计者的选择。

现在，我们介绍一下用 JSP 所编写页面的组成部分：

1）静态的 HTML/XML 部件。

2）专门的 JSP 标识（tag）。

3）用 Java 语言编写的程序片断。

因此，你可以用传统的 HTML/XML 工具软件来创建和编写 JSP 页面，同时还需要使用技术分离，所谓技术分离，就是说，页面的静态内容和动态内容是可以分开来写的。我们将和 JSP 的优点一起详细介绍这个技术原理。

（2）JSP 的发展历程。SUN 公司凭借自己在 Java 上的不凡造诣，继 Java 应用程序、Java Applet 之后，推出 Java Server Pages，这是其技术上的又一次创新。

JSP 的产生和发展历程是这样的：1998 年 4 月，SUN 公司发布 JSP 0.90 规范；1999 年 1 月，发布 JSP0.92 规范，同时推出支持 JSP 的 Web 服务器，从此 JSP 得到迅速发展；1999 年 11 月，SUN 发布 JSP 1.1 规范，同时推出 JSWDK 1.0.1、Servlet 2.2；2000 年 9 月，SUN 发布 JSP 1.2 规范和 Java Servlet API 2.3。

JSP 既然是 Java 家族的一员，自然也承袭了 Java 的所有优点，那就是跨平台性，JSP 并不限定在特定的操作平台或网络服务器上才能执行，因此，它给予网页设计者更大的发挥空间。

（3）JSP 的优点。

1）将内容的生成和显示进行分离。使用 JSP 技术，开发人员可以使用 HTML/XML 标识来设计最终页面，而使用 JSP 标识生成页面的动态内容。所生成的内容将被封装在 JavaBeans 组件中，并且捆绑在脚本语言中，所有的脚本语言在服务器端运行。如果核心逻辑被封装在 JSP 标识和 JavaBeans 中，那么 Web 管理人员和设计者能够编辑和使用 JSP 页面，而不影响内容的生成。在服务器端，这些标识和脚本的解释引擎会将从客户端发送过来的请求生成内容，将结果以 HTML/XML 的页面形式送回客户端浏览器。这就有利于 Web

的设计者保护代码版权，又保证了 Web 浏览器的安全性。同时 Web 页面设计人员编写 HTML，只需要留出地方让 Servlets 程序员插入动态部分即可。

2）可以重复使用组件。JSP 提出了可以在交互式的 Web 页面上重复使用组件的技术，也就是说“一次编写，多次使用”，JSP 页面可以更加容易的跨平台、跨 Web 服务器的移植，而不用改变任何条件来适应新环境。

3）采用标识简化页面开发。Web 页面开发人员不会都是熟悉脚本语言的程序员。JSP 简化了页面开发的工作量，它封装了许多在生成动态内容所必须的功能。这些功能通过接口来访问和实例化 JavaBeans 组件等等。因此，开发人员都可以为常用的功能来创建自己的标识库。这使得 Web 页面开发人员能够使用熟悉的工具来执行特定的功能。

### 2. JSP 的体系结构

（1）JSP 运行机制。JSP 的目的是为发展服务器端小应用程序提供一套方法。JSP 的规格说明已经作为一项标准定义在 Servlet API 的声明前部。具有代表性的是，JSP 运行在客户端向服务器端的请求时期和通信时期，通信时期只是执行一次，直到 JSP 页面被改变为止。假如我们的页面没有任何的语法错误，执行结果就生成执行服务器端的应用程序接口的类文件。JSP 的运行机制如图 1-2 所示。

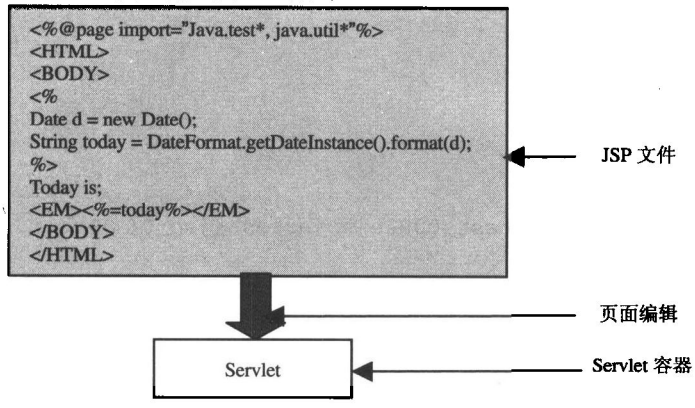



图 1-2　JSP 运行机制

当 JSP 第一次接收到有关页面请求的时候，它通过自己来执行整个通信时期的工作。请注意版本 JSP 1.1 的规范说明，它允许 JSP 页面预先编译成类文件，而预先编译则能够更加有效的消除 JSP 页面第一次接收来自客户端请求的启动延迟。而在这个通信时期的细节工作，例如源代码和类文件所要贮存的位置等则是由具体执行来决定。下面这个类文件是 JSP 页面例子（如图 1-2 所示），你不一定要完全看懂，但是最好能明白 JSP 到底是怎样去工作的。在通读了本书的实例之后，你就会明白本实例的执行流程。

```
package jsp;
//载入所需要的各种 java 类库
import javax.servlet.*;
import javax.servlet.http.*;
```

```
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.PrintWriter;
import java.io.IOException;
import java.io.FileInputStream;
import java.io.ObjectInputStream;
import java.util.Vector;
import org.apache.jasper.runtime.*;
import java.beans.*;
import org.apache.jasper.JasperException;
import java.text.*;
import  java.util.*;
public class _0005cjsp_0005cjsptest_0002ejspjsptest_jsp_0
//从父类 HttpJspBase 中继承
 extends HttpJspBase
{
 //声明静态变量
 static
 {
 }
 //声明公有变量
 public _0005cjsp_0005cjsptest_0002ejspjsptest_jsp_0( )
 {
 }
 //声明私有变量
 private static boolean _jspx_inited = false;
 //声明异常处理
 public final void _jspx_init() throws JasperException
 {
 }
 public void _jspService(HttpServletRequest request,
                         HttpServletResponse  response)
 throws IOException, ServletException
 {
    JspFactory _jspxFactory = null;
    PageContext pageContext = null;
    HttpSession session = null;
    ServletContext application = null;
    ServletConfig config = null;
```

```
JspWriter out = null;
Object page = this;
String  _value = null;
try
{
  //如果_jspx_inited为假，则开始初始化工作
  if (_jspx_inited == false)
  {
      _jspx_init();
      _jspx_inited = true;
  }
  _jspxFactory = JspFactory.getDefaultFactory();
  //设置响应内容为文本和html格式
  response.setContentType("text/html");
  pageContext = _jspxFactory.getPageContext(this, request,response, "",
                                            true, 8192, true);
  application = pageContext.getServletContext();
  config = pageContext.getServletConfig();
  session = pageContext.getSession();
  out = pageContext.getOut();
  out.write("\r\n<html>\r\n<body>\r\n");
  //从文件E:\\jsp\\jsptest.jsp的(3,2)开始到(5,0)
  //命名日期变量
  Date d = new Date();
  String today = DateFormat.getDateInstance().format(d);
  //输出界面内容
  out.write("\r\nToday is: \r\n<em> ");
  //从文件E:\\jsp\\jsptest.jsp的(7,8)开始到(7,13)
  //输出日期
  out.print(today);</b>
  out.write(" </em>\r\n</body>\r\n</html>\r\n");
}
catch (Exception ex)
{
  if (out.getBufferSize() != 0)
  //如果缓冲尺寸不为0，则清空
  out.clear();
  pageContext.handlePageException(ex);
}
```

```
        finally
        {
          out.flush();
          _jspxFactory.releasePageContext(pageContext);
        }
    }
}
```

这个 JSP 执行类文件和它的父类（HttpJspBase）轮流的运行 Servlet 接口。注意这个类用于服务的方法——_jspService()，它是 JSP 页面的本质内容。虽然_jspService()不能够被重载，但是开发者可以用它来初始化和销毁事件。这时，可以使用它所提供的方法 jspInit()和 jspDestroy()。一旦这个类文件被载到 servlet 容器内，这个_jspService()方法负责应答客户端所提出的请求。默认地，_jspService()为每一个客户端的请求通过 servlet 容器分配一个独立的线程。

JSP 的线程分配如图 1-3 所示。

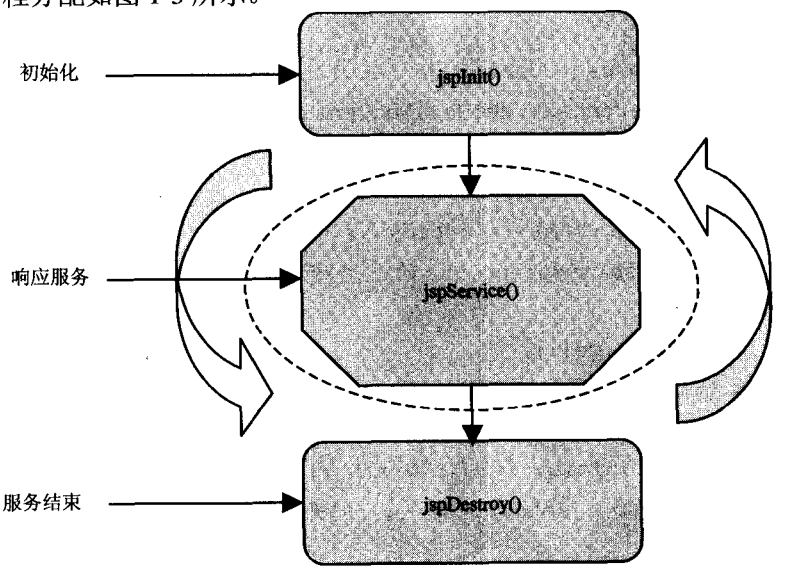



图 1-3　JSP 线程分配流程

（2）JSP 组织架构。在典型的 JSP 架构中，包含如下几个部分（图 1-4 为其关联图）。

1）客户端。基于浏览器客户端的应用程序比传统的基于客户机/服务器的应用程序有几个好处。这些好处包括几乎没有限制的客户端访问和极其简化的应用程序部署和管理（要更新一个应用程序，管理人员只需要更改一个基于服务器的程序，而不是成千上万的安装在客户端的应用程序）。这样，软件工业正迅速地向建造基于浏览器客户端的多层次应用程序迈进。

2）登录条目。登录条目、与数据库操作相联系的表单是大多数动态网页的主要内容，

通过提交这些表单触发对 JSP/Servlet 的执行（第一次使用时需要编译）。

3）服务器组件。

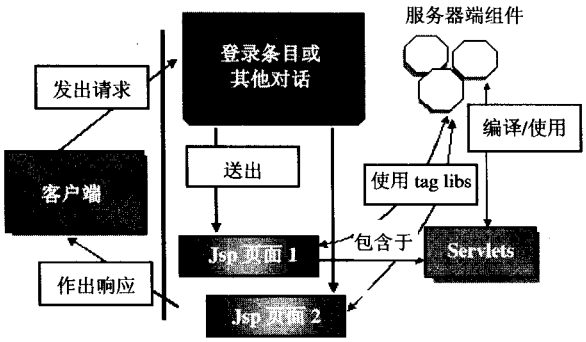



图 1-4　JSP 的典型架构

服务器端组件包括 JavaBeans、数据库、EJB 和其他第三方组件（如图 1-5 所示），JSP 包含一整套将这些组件有机的结合在一起的方法，而不论这些组件是基于何种平台的何种 Web 服务器。

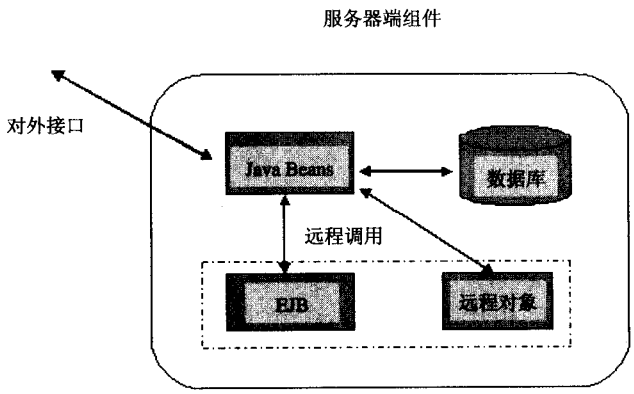



图 1-5　服务器端组件

绝大多数 JSP 页面依赖于可重用的、跨平台的组件(JavaBeans 或者 Enterprise JavaBeans 组件）来执行应用程序所要求的更为复杂的处理。开发人员能够共享和交换执行普通操作的组件，或者让这些组件为更多的使用者或者客户团体所使用。基于组件的方法加速了总体开发过程，并且使得各种组织在它们现有的技能和优化结果的开发努力中得到平衡。

（3）JSP 体系的两种模型。

1）模型 I：层叠的应用程序。如果在开发环境中所有的人员都熟悉 Java 和 HTML，或者这些工作自己能完成，则可以使用此模型。这个模型为每一个人员提供了明确的代码结构，而且，这个模型最大的优点是只要求维护一个文件。但最大的缺点是不易读，为调试和维护增加了难度。

下面的 Timezone.jsp 文件使用了模型 1，请注意其结构。

```
//使用的 XML 版本声明
<xml version="1.0" ?>
<H1>
  Time JSP
</H1>
<jsp:scriptlet>
//参数 zone 应该是从 0 到 24
TimeZone timeZone = TimeZone.getDefault();
//如果响应不为空值，则设置时间显示
if(request.getParameterValues("zone") != null)
{
    String timeZoneArg = request.getParameterValues("zone")[0];
    timeZone = TimeZone.getTimeZone("GMT+" + timeZoneArg + ":00");
}
Calendar myCalendar = Calendar.getInstance(timeZone, Locale.UK);
//直接显示时间
</jsp:scriptlet>
<%= myCalendar.get(Calendar.HOUR_OF_DAY) %>:
<%= myCalendar.get(Calendar.MINUTE) %>:
<%= myCalendar.get(Calendar.SECOND) %>
```

2）模型 II：改变请求路径。如果在开发环境中有的人员熟悉 HTML，有的人员熟悉 Java 编程，则适合使用此模型。Java 程序员可以致力于创建可重用的代码，而 HTML 设计人员则致力于页面的设计。两者之间相互依赖不大，这就方便于修改与维护。

现在使用模型 II 来代替模型 I 对上面的例子进行重写。例子中使用了一个类来处理请求、取得时区和设置所需的变量值等。这样的方法可以被称作“模型－视图－控制”（MVC）方法。

使用模型 II 编写的显示时间模型的 timeByZone.jsp 文件如下。

```
//XML 版本声明
<xml version="1.0" ?>
<!--Worker Class, nobody should see me-->
<jsp:scriptlet>
//参数 zone 应当取 0－24 之间的数值（包括 0 和 24）
TimeZone timeZone = TimeZone.getDefault();
//返回默认的 TimeZone
if(request.getParameterValues("zone") != null)
```

```
{
    String timeZoneArg = request.getParameterValues("zone")[0];
    timeZone = TimeZone.getTimeZone("GMT+" + timeZoneArg + ":00");
}
TimeBean timeBean   = new TimeBean();
timeBean.setHours   = myCalendar.get(Calendar.HOUR_OF_DAY); //获得小时
timeBean.setMinutes = myCalendar.get(Calendar.MINUTE); //获得分钟
timeBean.setSeconds = myCalendar.get(Calendar.SECOND); //获得秒
HttpSession mySession = request.getSession();
mySession.putValue("tempTimeBean", timeBean);
</jsp:scriptlet>
<jsp:forward page="displayTime.jsp" />
```

显示时间视图的 displayTime.jsp 文件如下。

```
<xml version="1.0" ?>
//设置显示格式
<H1>
 Time JSP
</H1>
//JSP 属性设置
<jsp:useBean class="TimeBean" id="tempTimeBean" scope="session" />
<jsp:getProperty name="tempTimeBean" property="hours">:
<jsp:getProperty name="tempTimeBean" property="minutes">:
<jsp:getProperty name="tempTimeBean" property="seconds">
//如果没有被实例化则打印"null"
<jsp:scriptlet>
HttpSession mySession = request.getSession();
TimeBean  timeBean  = mySession.getValue("tempTimeBean");
//检查是不是空值，避免抛出 NullPointerExceptions 异常
if(timeBean != null)
{
    //输出显示小时
    out.print(timeBean.getHours());
    out.print(":");
    //输出显示分钟
    out.print(timeBean.getMinutes());
    out.print(":");
    //输出显示秒
```

```
    out.print(timeBean.getSeconds());
}
else
{
    //如果是空值，则产生如下提示，让用户返回
    out.println("Press your Back button and select a TimeZone");
}
</jsp:scriptlet>
```

### 3. JSP 与其他动态网页技术的比较

（1）动态 Web 编程简介。静态 HTML 对于显示相对静态的内容是不错的选择。新的挑战在于创建交互的基于 Web 的应用程序，在这些程序中，页面的内容是基于用户的请求或者系统的状态，而不是预先定义的文字。

对于这个问题的一个早期解决方案是使用 CGI-BIN 接口。开发人员编写与接口相关的单独的程序，以及基于 Web 的应用程序，后者通过 Web 服务器来调用前者。这个方案有着严重的扩展性问题——每个新的 CGI 要求在服务器上新增一个进程。如果多个用户并发访问该程序，这些进程将消耗该 Web 服务器所有的可用资源，并且系统性能降低到极低的地步。

某些 Web 服务器供应商已经尝试通过为它们的服务器提供“插件”和 API 来简化 Web 应用程序的开发。这些解决方案是与特定的 Web 服务器相关的，不能解决跨多个供应商的解决方案的问题。而且其开发难度很大，一般的编程者很难利用 CGI、ISAPI 编出实用的 Web 应用程序来。

随后，替代 CGI，ISAPI 的解决方案纷纷出笼，有 Mircrosoft 的 ASP、Tcx 的 PHP（Personal Homepage Tools）和 SUN 的 JSP 等。在这诸多方案中，JSP 以其诸多优越性，渐渐成为其中的佼佼者，大有后来居上的势头。

（2）JSP 与 CGI 及 ISAPI 的比较。图 1-6 显示了 JSP 与 CGI 及 ISAPI 的比较结果。

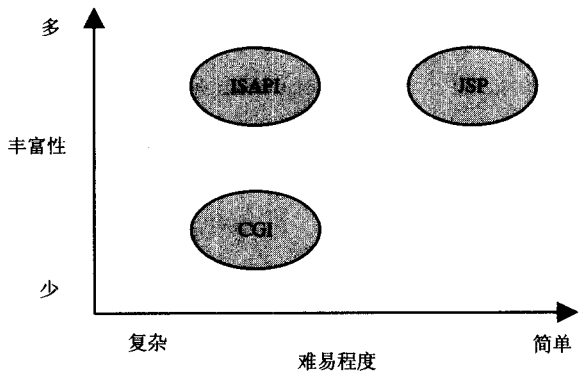



图 1-6　JSP 与 CGI、ISAPI 的对比

CGI即Common Gateway Interface，它几乎被所有Web服务器支持，但随着技术的发展，它渐渐的失去了原有的重要位置。在安装UNIX操作系统的计算机上，多数CGI应用程序都是用Perl（Practical Extraction And Report Language）编写的。C语言也是创建CGI的常见选择。从开发角度来看CGI程序最大的缺点在于它很难调试与维护。从性能来看，由于CGI对每个请求都产生一个新进程，所以在访问量较大时，它要使用大量资源并浪费许多时间。CGI一个后继者是ISAPI（Internet Server API），后者克服了CGI为每个请求建立新进程的缺点，转而用DLL（动态链接库）来实现，这样无疑加快了应用程序的执行速度。但是这种实现的难度也很大，而且拙劣的DLL可能会导致Web服务器的崩溃。但是，正如我们所看到的那样，“青出于蓝而胜于蓝”，在不断的技术更新中，ASP、PHP和JSP先后涌现出来，并取得了后来居上的位置。JSP在同CGI、ISAPI性能和编程的难易程度的对比中都处于优越的位置（如图1-6所示）。

（3）JSP和ASP比较。由微软公司推出的ASP全名是Active Server Pages，是一个Web服务器端的开发环境，利用它可以产生和运行动态的、交互的、高性能的Web服务应用程序。ASP采用脚本语言VB Script或Java Script作为自己的开发语言。ASP技术使得在Web页面上创建动态内容更加容易，但是也只能工作在微软的IIS（Microsoft Internet Information Server）和PWS（Personal Web Server）上。

从名字来看JSP与ASP就很相似。而事实上，微软的ASP和SUN公司的JSP在技术方面的确有许多相似之处。两者都是为基于Web应用实现动态交互网页制作提供的技术环境支持，都能够为程序开发人员提供实现应用程序的编制与自带组件设计网页从逻辑上分离的技术，而且都能够替代CGI使网站建设与发展变的较为简单与快捷。

尽管JavaServer Pages技术和微软的Active Server Pages在许多方面都有相似之处，但仍然存在很多不同，其中最本质的区别在于：两者是来源于不同的技术规范组织，其实现的基础，以及Web服务器平台要求也不相同。

具体的，让我们从不同角度来将JSP与ASP作一番比较。

1）难度。JSP与ASP都是简单易学尤其是易入门的语言。相比之下，ASP似乎被认为更容易一些。读者如果已经掌握了HTML页面开发，并对SQL类型的数据库有一定了解，那就可以很容易的、快速的学习和理解ASP。同样的，对于掌握了Java编程和JDBC的读者，JSP也是向动态Web编程进军的最简单有力的工具。

2）结构。无论是JSP页还是ASP页，都不过是一些插入了服务器端脚本代码的HTML文件。JSP文件的扩展名是.jsp、ASP文件的扩展名是.asp，而不是.htm或.html，以便对宿主Web服务器指示服务器端代码。无论JSP文件还是ASP文件，都是文本文件，可以用最简单的文本工具Notepad（记事本）打开并编辑——当然，Notepad并不适宜作为JSP、ASP的开发工具，ASP最常见的开发环境是Windows+InterDev，JSP环境和开发工具将在后面章节介绍。

JSP与ASP在结构上非常相似，都是以“<%”和“%>”作为标识符。不同的是在标识符之间的代码ASP为JavaScript或VBScript脚本，而JSP为Java代码。下面的例子中，我们分别用JSP和ASP实现输出显示当前时间的动态页面，读者可以从该程序的代码结构和各自的运行效果两方面将二者做个比较。

JSP文件date.jsp如下所示。